Python Fun(damentals)

Python Variables

Alexander Rymdeko-Harvey

Obscurity Labs

```
* Innovation
* Expert Training
* Advanced Security Services
```

Introduction

- Who am I?
- Why this is important?
- Free training?

Install Demo and Instructions

1. Windows/Mac/Ubuntu Install Instructions
2. Python 3.6+ & Packages
3. Visual Studio Code
4. Pipenv Instruction

Python Variables

Its important to understand C/C++ principles first, to understand what makes Python variables so easy to work with. Lets take the following example in C:

```c
/* variable definition: */
char str[50] = "Better know what im doing";
```

Now in Python, notice we dont declare it as a char or need size?

```python
a = 'Yea.. no size here'
```

Python Variable Memory

The previous example works because:

- Python variables are nothing but reserved memory locations
- Based on variable type the Python interpreter allocates memory
- Declaration happens automatically when you assign a value to the variable
- This can easily be changed on the fly, expanded or shrunk
- This is the beauty of an object oriented non-statically typed language.

Python Variable Memory Cont.

Here is a example of a C program accessing memory ptr locations.

```
#include <stdio.h>
#include <stdlib.h>
void main(void)

{
int var = 34;
int *ptr;
ptr = &var;
printf("\nDirect access,
        variable var value = var = %d", var);
printf("\nIndirect access,
        variable var value = *ptr = %d", *ptr);
printf("\n\nThe memory
        address of variable var = &var = %p", &var);
printf("\nThe memory
        address of variable var = ptr = %p\n", ptr);
}
```

Python Variable Memory Cont.

Here is a snipt of Python code I used to access `ptr` locations in memory:

```
>>> a = 'alex'
>>> id(a)
4477448064
>>> hex(id(a))
'0x10ae06f80'
```

`id(object)`

Return the “identity” of an object. This is an integer which is guaranteed to be unique and constant for this object during its lifetime. Two objects with non-overlapping lifetimes may have the same id() value.

So what?

1. The Python implementation should not be tied to a particular platform. It's okay if some functionality is not always available, but the core should work everywhere.
2. Since most modern OS are written in C, compilers/interpreters for modern high-level languages are also written in C. Python is not an exception - its most popular/"traditional" implementation is called **CPython and is written in C**
3. PEP 20 -- The Zen of Python - https://www.python.org/dev/peps/pep-0020/

Python Variable Declaration

Here is some string simple example:

```
>>> a = '1'
>>> b = '2'
>>> ab = a + b
>>> print(ab)
12
```

Here is some string int example:

```
>>> a = 1
>>> b = 2
>>> ab = a + b
>>> print(ab)
3
```

Notice what happen here?

Python Variable Declaration Cont.

Here is example using the `os` module in Pythons core.

```
>>> import os
>>> currentPid = os.getpid()
>>> print(currentPid)
45034
```

- Variables can be used to capture module results
- Documentation is key to understanding what a module Returns
- This allows proper use

Python Variable Documentation

Here the documentation for the OS module

https://docs.python.org/3/library/os.html

```
os.getpid()
        Return the current process id.
```

Lab_1.py

TASKING

Take the time to create two key variables, `currentProgram` which is a string that you define of the current program name. Second the use the python builtin os module to get current process id (PID) and declare it as `currentPid`. print both to the console using any method you would like to use.

Lab_2.py

TASKING

Take the time to create two variables that will be used in the future projects. import the module `sys` and use it to get the current modules loaded in the current namespace for debug purposes. Please name this variable `systemMod` and than convert this variable to a string which will be placed into `systemModS`.